WROCŁAW UNIVERSITY OF SCIENCE AND TECHNOLOGY
Master’s in Computer Engineering – Information Systems Modeling Laboratory

Lab 04 Report – Frontend & AOP

Name: Mohammed Altarhouni
Student ID: 288739

1. Introduction

This report documents the combined work for the Sushi Bar REST API and the FoodApp frontend completed in Lab 04. The objective was to extend the backend system with aspect-oriented programming (AOP) for statistics collection and develop a basic frontend interface that demonstrates CRUD operations using the REST API.

2. Sushi Bar REST API

The backend is a Java 17 Spring Boot application composed of two modules:

- Sushi-Bar-api: Contains OpenAPI-generated DTOs and models.

- Sushi-Bar-services: Contains controllers, services, data layer, and business logic.

Key Features:

- Supports endpoints for menu management, orders, reservations, user accounts, and authentication.
- OpenAPI specification (sushi-bar.yaml) defines models with pagination, filtering, and error handling.
- Semantic Web features are showcased with RDF/RDFS samples.
- Includes a custom statistics endpoint for runtime metrics.

2.1 Aspect-Oriented Features

- A custom aspect @LogAndMeasure intercepts all controller calls.
- Captures method call counts and the highest price returned.
- Tracks the most frequently invoked HTTP method and path combination.
- All statistics are stored in memory and exposed via the /statistics endpoints.

2.2 Statistics Controller Endpoints

Endpoint	Description
GET /statistics/summary	Aggregate method usage and percentage breakdown
GET /statistics/methods	Detailed list of methods with counts and timing
GET /statistics/methods/top/{n}	Top N most frequently used methods

GET /statistics/methods/paginated	Paginated access to method statistics
GET /statistics/dashboard	Dashboard with total calls, highest values, and summary

3. FoodApp Frontend

The frontend is a lightweight web application developed using Angular. It interacts with the Sushi Bar REST API through auto-generated TypeScript services.

Main Features:
- Displays menu items with options to create and view orders.
- Implements reactive forms and Angular routing.
- No authentication is required for lab purposes.
- Compatible with any modern JavaScript framework.

4. Building & Running

4.1 Backend (Sushi Bar)

```
./mvnw clean install
./mvnw spring-boot:run -pl Sushi-Bar-services
```

4.2 Frontend (FoodApp)

```
npm install
npm start
```

Swagger UI: http://localhost:8080/swagger-ui.html
Angular App: http://localhost:4200

5. Example Workflow

1. Add menu items via Swagger UI or the frontend.
2. Place an order through the FoodApp interface.
3. Access /statistics/dashboard to see statistics like most-used endpoints and highest price.

Example JSON Output:

```
{
  "methodStats": [
    {"methodKey": "GET /menu", "callCount": 5, "maxReturnedPrice": 25.99},
    {"methodKey": "POST /menu", "callCount": 2, "maxReturnedPrice": 18.50}
  ],
  "mostCalledMethod": "GET /menu",
  "mostCalledCount": 5,
  "highestPrice": 25.99
}
```

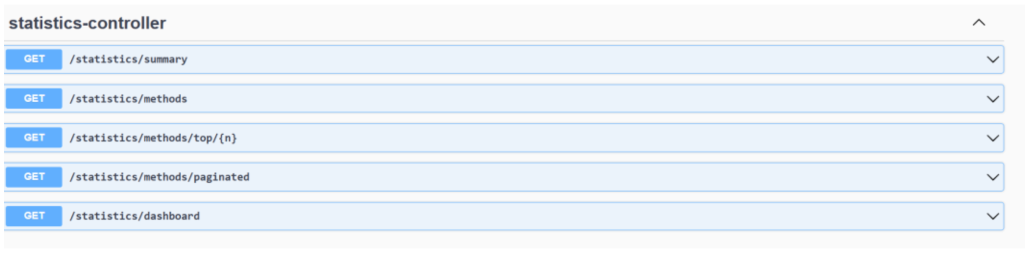
6. Deliverables

- Source code of both backend and frontend projects (excluding build artifacts and node_modules)
- This report in PDF or DOCX format, including setup steps and screenshots
- An optional demo video showing system functionality

7. Remarks

- Only selected views from the SRS were implemented to maintain focus
- Aspect-Oriented Programming allowed seamless metric gathering without modifying controllers
- Future improvements could include:
  - Persisting statistics to a database
  - Adding authentication to the frontend

8. Screenshots


statistics-controller

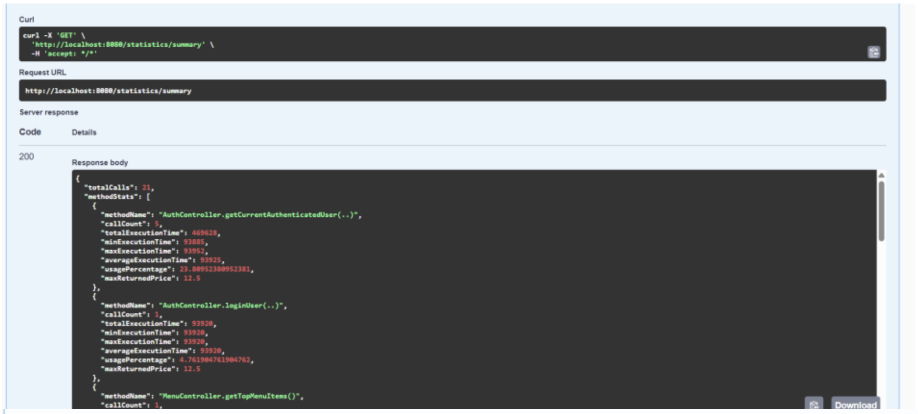
GET /statistics/summary

GET /statistics/methods

GET /statistics/methods/top/{n}

GET /statistics/methods/paginated

GET /statistics/dashboard


Curl

```
curl -X 'GET' \
  'http://localhost:8080/statistics/summary' \
  -H 'accept: */*'
```

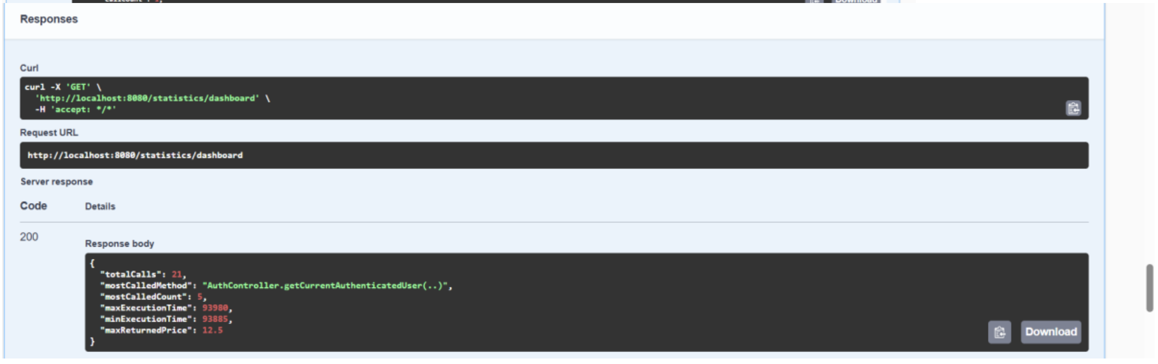
Request URL

```
http://localhost:8080/statistics/summary
```

Server response

Code	Details
200	Response body

```
{
  "totalCalls": 21,
  "methodStats": [
    {
      "methodName": "AuthController.getCurrentAuthenticatedUser(..)",
      "callCount": 5,
      "totalExecutionTime": 469628,
      "minExecutionTime": 93885,
      "maxExecutionTime": 93952,
      "averageExecutionTime": 93925,
      "usagePercentage": 23.80952380952381,
      "maxReturnedPrice": 12.5
    },
    {
      "methodName": "AuthController.loginUser(..)",
      "callCount": 1,
      "totalExecutionTime": 93920,
      "minExecutionTime": 93920,
      "maxExecutionTime": 93920,
      "averageExecutionTime": 93920,
      "usagePercentage": 4.761904761904762,
      "maxReturnedPrice": 12.5
    },
    {
      "methodName": "MenuController.getTopMenuItems()",
      "callCount": 1,
```


Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/statistics/dashboard' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/statistics/dashboard
```

Server response

Code	Details
200	Response body

```
{
  "totalCalls": 21,
  "mostCalledMethod": "AuthController.getCurrentAuthenticatedUser(..)",
  "mostCalledCount": 5,
  "maxExecutionTime": 93980,
  "minExecutionTime": 93885,
  "maxReturnedPrice": 12.5
}
```